Graph Algorithms - Lecture 1

October 5, 2018

Table of contents

1. Course description
   - Course page and related materials
2. Graph theory applications
3. Graph Theory Vocabulary
   - Graph Definition
4. Exercises for the next week seminar

Course page and related materials

Course page: http://profs.info.uaic.ro/~olariu

Objectives:
The lectures will cover basic topics in Algorithmic Graph Theory. Accumulated knowledge will be applied in designing efficient algorithms for combinatorial optimization problems

Teaching methods:
Video presentations of the lecture notes, which will be posted as pdf files before each lecture. These notes will contain the exercises for the seminars.

Seminars: E. F. Olariu, A. C. Frăsinaru, A. Policiuc, V. Motroi

Each seminar debates a number of exercises (posted in advance in the lecture notes) in order to deepen the subjects introduced in the lectures. Students are encouraged to find original solutions.

Grading:

- Seminar activity: attendence (max. 8 points), participation (max. 10 points) - max. 18 points.
- Homeworks: three exercise sheets, max. 14 points each - max. 42 points
- Written final test: max. 60 points

From a maximum of 120 points the threshold for promovating the course is of 50 points.

Course page and related materials

Outline of the course:

- Vocabulary of Graph Theory.
- Path problems: graph traversal, shortest-paths, connectivity.
- Minimum spanning trees: union-find, amortized complexity.
- Matching theory.
- Flows.
- Polynomial time reductions between decision problems on graphs.
- Approaching **NP**-difficult problems.
- Planar graphs.

Graph theory applications

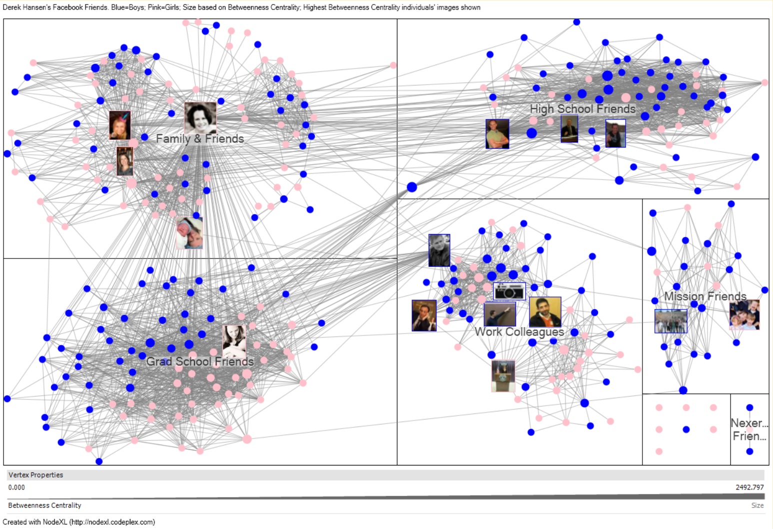


Figure: Facebook/Twitter

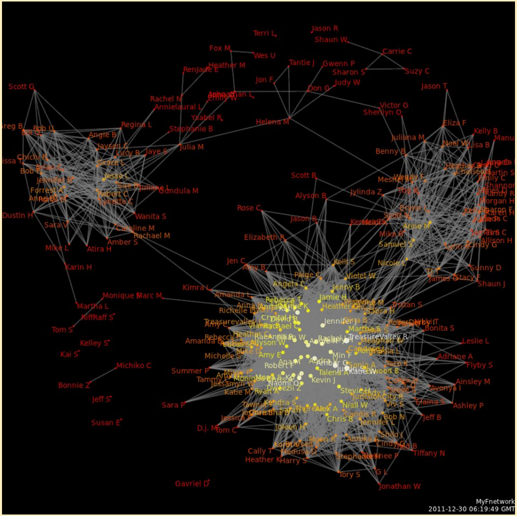


Figure: A Facebook network of a few users

Graphs are used for analyzing social/news networks (like Facebook or Twitter) in order to find characteristics like:

- the level of connectivity, the density;
- the influence of users within the network (centrality, the social networking potential);
- the level of segmentation: measuring the clusterization;
- robustness or structural stability of the network.

The network analysis is then used for

- data mining and aggregation, marketing;
- behavior analysis and network prediction;
- intelligence and security analysis (surveillance of terrorism and organized crime) etc.

Apart from social network study which is a hot topic nowadays there are a lot of other applications for graph algorithms:

- **Minimum spanning tree**: for efficiently connecting communicating points (e. g. in IT&C);
- **Eulerian paths and circuits**: The chinese postman problem - find a shortest closed path/circuit that visits once every edge of a graph (for street cleaning, mail/services/packages delivery, garbage collection etc.);
- **Hamiltonian paths and circuits**: efficiently visiting once a number of points (in a city, a country etc); Travelling salesman problem, Vehicle routing problem;

- **Graph colorings**: coloring maps (faces of planar graphs), lectures/seminars/timetabling, exam scheduling for university departments, mobile radio frequency allocation, memory register allocation.

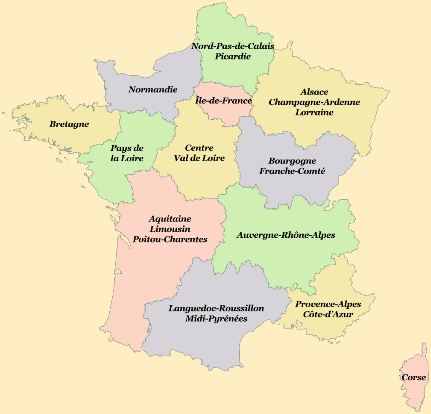


Figure: France regions from 2016

Graph theory applications

- Matchings: assignment problems, in computational chemistry and mathematical chemistry studies on organic materials.

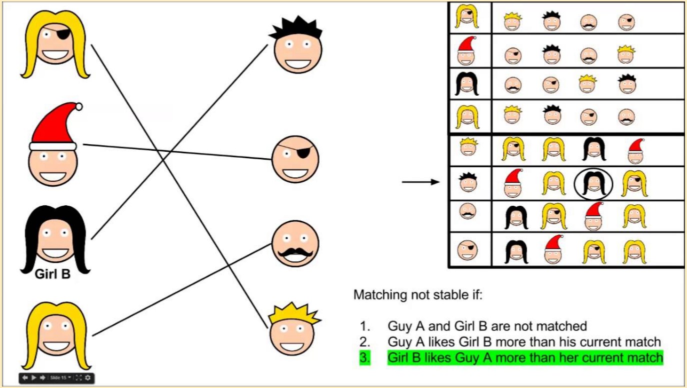


Figure: Gale Shapley algorithm

- **Stable matchings**: assignment medical students to hospitals, the admission procedure of higher education institutions, identifying optimal assignment of kidney (in transplantations), student-project allocation problem etc.
- **Maximum flow**: find the charging level in transportation networks for improving the traffic parameters, image reconstruction from X-Rays projections (in tomography), scheduling on parallel processors.

Applications in Computer Science

- In database management, graph database uses graph-structure data for storing and querying.
- Graph rewriting systems are used in software verification.
- Quantum computation.
- Modeling the web documents as graphs and clustering of web documents.
- Approximation of data and data compression.
- Modeling the sensor network as graphs (using Voronoi diagrams).

Notations

For a given finite set X:

- $|X| = card(X) \in \mathbb{N}$ is the cardinality of X;
- If $|X| = k$, then X is a k-set;
- $2^X = \mathcal{P}(X)$ is the power set of X: $2^X = \{Y : Y \subseteq X\}$, $|2^X| = 2^{|X|}$;
- $\binom{X}{k} = \{Y : Y \subseteq X, |Y| = k\}$, $\left|\binom{X}{k}\right| = \binom{|X|}{k}$.

Definition 1

A **graph** *is a pair* $G = (V, E)$, *where:*

- $V = V(G)$ *is a finite, non-empty set; is the set of* **vertices** (**nodes**) *of* G;
- $E = E(G)$ *is a subset of* $\binom{V(G)}{2}$; *is the set of* **edges** *of* G.

$|G| = |V(G)|$ *is the* **order** *of graph* G, *and* $|E(G)|$ *is the* **size** *of* G.

Definition 2

If $G = (V, E)$ and $e = uv = vu = \{u, v\} \in E$ is an edge of G, then we say that

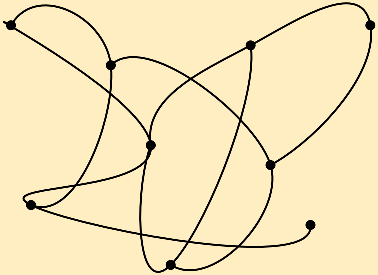
- *e* **connects** *(or* **links***) vertices u and v; that*
- *vertices u and v are* **adjacent** *or u and v are* **neighbors***;*
- *e is* **incident** *with u and v;*
- *u and v are the* **endpoints** *(***extremities***) of e.*

Neighborhood of vertex u is $N_G(u) = \{v \in V(G) : uv \in E(G)\}$.

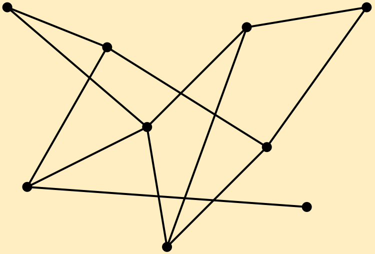
Two edges e and f are **adjacent** if they share a common endpoint: $|e \cap f| = 1$.

A graph can be represented in plane as a figure consisting of a set of nodes (small geometric forms: points, circles, squares etc) corresponding to its vertices and curves which connect the vertices corresponding to the edges from the graph.

An example:



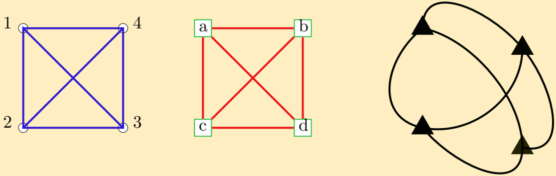
The same graph again:



We can add labels (names, numbers etc) and colors to vertices and edges obtaining better visual representations.

Below there are three visual representations of the same graph:

$$G = (\{1, 2, 3, 4\}, \{12, 13, 14, 23, 24, 34\})$$



Definition 3

Stable set (or independent set of vertices) in $G = (V, E)$: $S \subseteq V$ such that $\binom{S}{2} \cap E = \varnothing$.

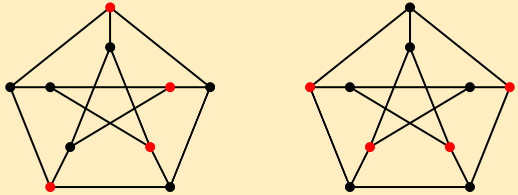
In other words, $S \subseteq V$ is a stable set of G if there is no edge between its vertices.

Notation

The maximum cardinality of a stable set of G is the stability number (or independence number) of G and is denoted by $\alpha(G)$.

Example

In the following graph (Petersen's) we have two different stable sets of maximum (why?) cardinality:



Stable sets: Optimization problem

P1 Input: G graph.
Output: $\alpha(G)$ and a witness stable set S with $|S| = \alpha(G)$.

Stable sets: Decision problem

SM Instance: G graph, $k \in \mathbb{N}$.
Question: Is there a stable set S in G, such that $|S| \geqslant k$?

NP-complete (Karp, 1972).

Definition 4

Matching (independent set of edges) in $G = (V, E)$: $M \subseteq E$ such that for all $e, f \in M$, if $e \neq f$, then $e \cap f = \varnothing$.

In more words, $M \subseteq E$ is a matching if each pair of its edges share no endpoint.

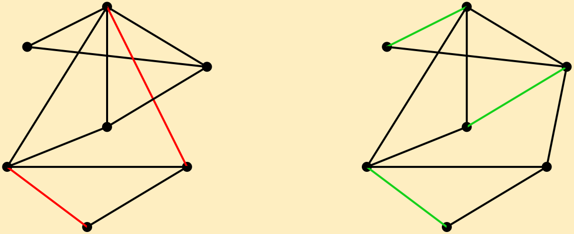
Notation

The maximum cardinality of a matching in G is called the **matching number** (**edge-independence number**) of G and is denoted by $\nu(G)$.

Graph Definition

Example

For the following graph two matchings are depicted with red and green (the second being of maximum - why? - cardinality):



Maximum matching: **Optimization problem**

P2 Input: G graph.
Output: $\nu(G)$ and a witness matching M with $|M| = \nu(G)$.

Edmonds (1965) showed that $P2 \in$ P.

Note

The problems $P1$ and $P2$ are similar: in both we are required to find a member of maximum cardinality of a family of sets concerning a given graph. What makes the difference?

Definition 5

For $p \in \mathbb{N}$, a p-coloring *of the graph $G = (V, E)$ is a map $c : V \to \{1, \ldots p\}$ such that $c(u) \neq c(v)$ for each $uv \in E$.*

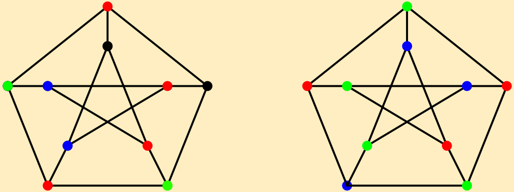
It is worthnoting that the set of all vertices having the same given color is a stable set (it is called also a coloring class). Since adjacent vertices have different colors, a p-coloring is a partition of V with at most p stable sets (or coloring classes).

Notation

The chromatic number of the graph G is the least value of p such that G has a p-coloring. This parameter is denoted by $\chi(G)$. ($\chi(G) \leqslant |G|$ - why?)

Example

For the following graph we depicted two colorings ($\chi(G) = 3!$)



Vertex coloring: Optimization problem

P3	Input:	G graph.
	Ouput:	$\chi(G)$ and a witness $\chi(G)$-coloring.

Vertex coloring: Decision problem

COL	Instance:	G graph, $p \in \mathbb{N}$.
	Question:	Is there a p-coloring of G?

NP-complete for $p \geqslant 3$ (Karp, 1972).

Definition 6

*For $p \in \mathbb{N}$, a **p-edge coloring** of the graph $G = (V, E)$ is a map $c : E \to \{1, \ldots p\}$ such that $c(e) \neq c(f)$ for all $e, f \in E$ with $|e \cap f| = 1$.*

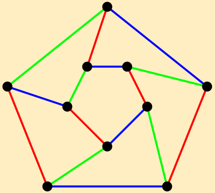
We can note that in a p-edge coloring a set of edges with the same color is a matching. Hence, a p-edge coloring is a partition of E with at most p matchings.

Notation

Chromatic index of the graph G: the least value of p such that G has a p-edge coloring. This parameter is denoted by $\chi'(G)$. ($\chi'(G) \leqslant |E(G)|$ - why?)

Example

For the following graph we depicted an edge coloring ($\chi'(G) = 3$!)



Edge coloring: Optimization problem

P3 Input: G graph.
Ouput: $\chi'(G)$ and a witness $\chi'(G)$-coloring.

Edge coloring: Decision problem

COL Instance: G graph, $p \in \mathbb{N}$.
Question: Is there a p-edge coloring of G?

NP-complete for $p \geqslant 3$ (Holyer, 1984).

Definition 7

Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are **isomorphic** *if there is a bijection $\varphi : V_1 \to V_2$ such that for every two vertices $u_1, v_1 \in V_1$, u_1 and v_1 are adjacent in G_1 (i. e., $u_1v_1 \in E_1$) if and only if $\varphi(u_1)$ and $\varphi(v_2)$ are adjacent in G_2 (i. e., $\varphi(u_1)\varphi(u_2) \in E_2$).*

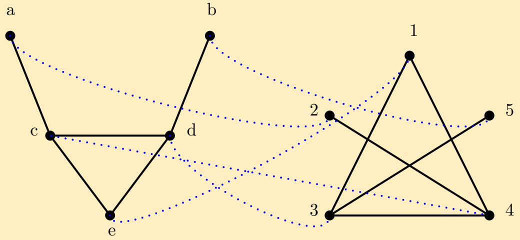
In other words two graphs are isomorphic if there is a bijection between their sets of vertices which induces a bijection between their sets of edges.

Notation

$G_1 \cong G_2$.

Example

The graphs from below are isomorphic.



Graph Definition

Isomorphism testing: **Optimization problem**

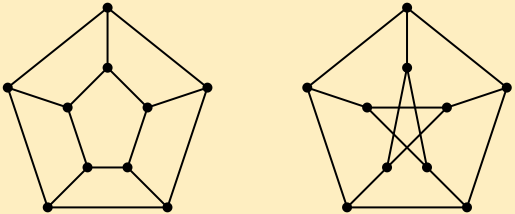
ISO	Input:	graphs G_1 and G_2.
	Ouput:	are G_1 and G_2 isomorphic?

Neither known to be in P nor NP-complete.

There is a quasipolynomial time algorithm (i. e., running in $2^{\mathcal{O}((\log n)^c)}$ for some $c > 0$, Babai, 2015).

Example

The following two graphs have the same order, size and degree sequence, but they are not isomorphic (why?).



Exercise 1. For $k \in \mathbb{N}^*$, set $G_d = K_2 \times K_2 \times \ldots \times K_2$ (k times).

(a) Find the order and the dimension of G_k

(b) Show that G_k is bipartite and determine $\alpha(G_k)$.

Exercise 2.

(a) Is there a graph with degrees $3, 3, 3, 3, 5, 6, 6, 6, 6, 6, 6$?

(b) Is there a bipartite graph with degrees $3, 3, 3, 3, 3, 5, 6, 6, 6, 6, 6, 6, 6, 6$?

(c) Is there a graph with degrees $1, 1, 3, 3, 3, 3, 5, 6, 8, 9$?

Exercise 3.

Two students, L(azy) and T(hinky), must find a particular path between two given vertices in a very sparse graph G: $|E(G)| = O(|V(G)|)$. L judge that, since the graph is sparse, the number of paths between the two vertices is small, and a lazy solution is to generate (backtracking) all these paths and than retain the wanted one. T does not agree and gives the following example: let $H = K_2 \times P_{n-1}$ ($n \geqslant 3$); add to H two new vertices x and y each joined by two edges with one of the two pairs of adjacent vertices in H of degree 2.

The obtained graph, G, is sparse but the number of paths between x and y is big. Explain to L this example by drawing G, showing its sparsity, and finding the number of paths between x and y.

Exercise 4. An examination session must be scheduled based on the following input specifications: the set of exams is known; each student sends the list of exams to which (s)he is registered; each exam take place with all students registered to it (written exam); each student can participate to at most one exam in the same day.

Construct a graph in order to answer the following questions (by determining appropriate parameters):

(a) What is the maximum number of daily exams?

(b) What is the minimum number of days for the examination session?

Exercise 5. (exercise 4 cont'd)

A smart and skilled programmer is wondering if the two NP-hard problems in the above exercise can be solved in polynomial time since the graph constructed seems to belong to a very special class of graphs.

(a) Prove that for any graph, G, there is an input for the scheduling problem above such that the graph constructed is G.

The programmer suggests the following "greedy approach" to solve the second question in exercise 4: starting with day 1, a maximum number of exams are scheduled (from the set of exams not scheduled) daily, until all exams are scheduled.

b) Show that this greedy strategy can fail, by giving a counterexample.

Exercise 6. A compiler optimization is the register allocation technique: the most used variables are kept in the fast processor registers in order to have them there at the moment when the compiler needs them (for certain operations made by CPU).

Design a graph that - by using its appropiate parameters - will answer to the following questions:

(a) What is the maximum number of variables which are not needed at the same time?

(b) What is the minimum number of registries needed?

Hint: We have two kind of objects at hand: variables (with their values) and CPU operations (or operators) which use one or more variables.

Exercise 7. (exercise 6 cont'd) A student is wondering if the above questions (which correspond to NP-hard problems) can be answered by polynomial time algorithms since the graph constructed seems to belong to a very special class of graphs.

(a) Prove that for any graph G there is an input for the registry allocation problem such that the resulted graph is G.

The student suggests the following "greedy approach" to answer the second question in exercise 6: starting with the first step, allocate as many as possible variables to one new free register in each step, until there are no more variables.

(b) Show that this greedy approach fails, by giving a counterexample.

Exercise 8. G is called *autocomplementary* graph if G and its complement, $\overline{G}$, are isomorphic ($G \cong \overline{G}$).

(a) Show that an *autocomplementary* graph is connected and $|G| \equiv 0$ or 1 *mod* 4.

(b) Find all *autocomplementary* graphs with at most 7 vertices.

(c) Show that, for every graph G, it exists an *autocomplementary* graph H, such that G is an induced subgraph in H,